Session 706

Debugging on Mac OS X

Dave Payne
Manager, Developer Applications Team

What You'll Learn

- Analyzing crashed applications
- Examining hung applications
- Debugging with Project Builder
- Advanced debugging with GDB
- Less common debugging situations

Why Did My App Crash?

- Add this to /etc/hostconfig
  CRASHREPORTER=-YES-
- Crash logs appears in:
  - crash.log window in Console application
  - /var/vm/crash.log
  - Caution—output retained across reboots!
- Logs include:
  - Stack backtrace for all threads
  - Register state for thread that crashed

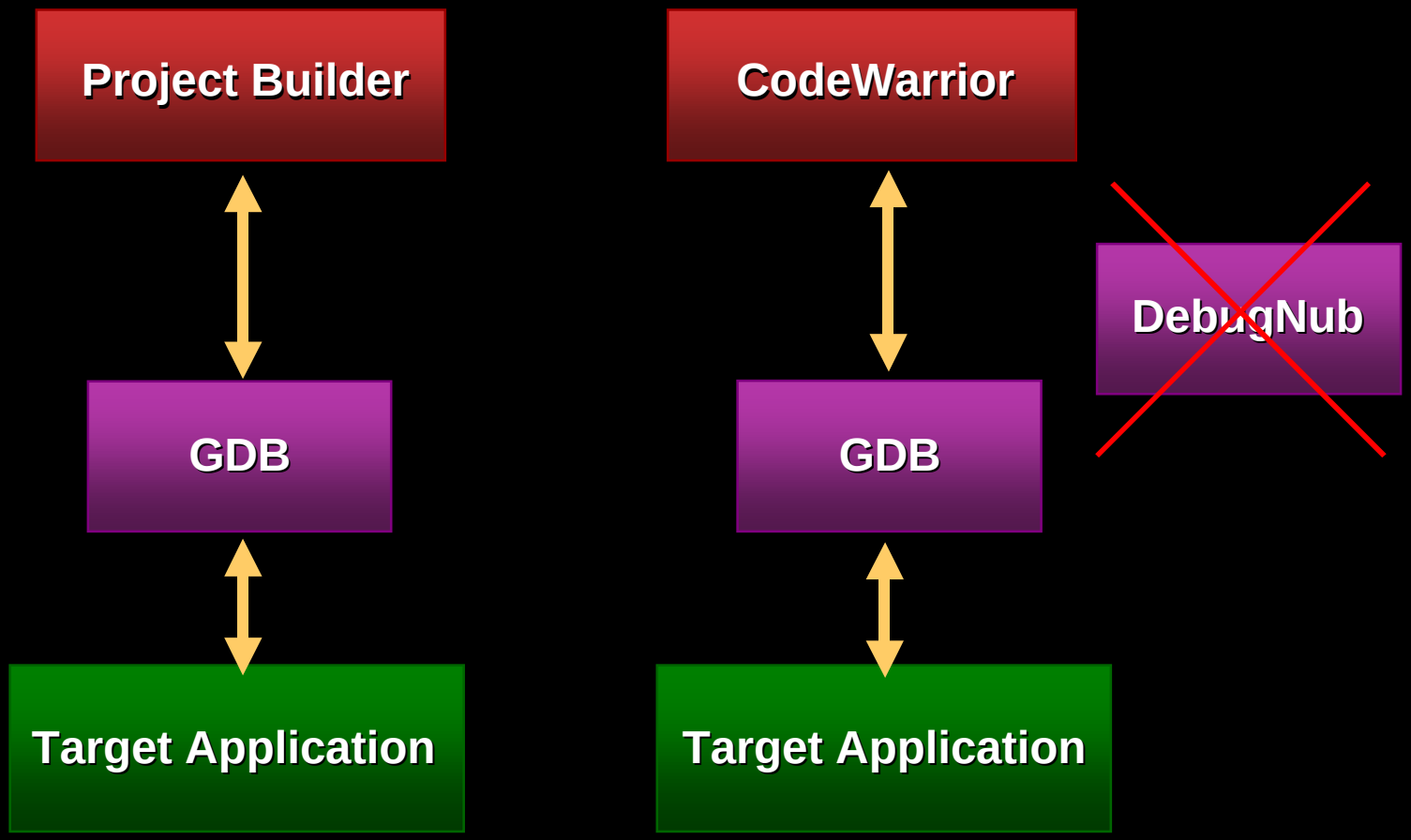
Why Is My App Hung?

- Use ‘sample’ command line tool
  - Command line version of Sampler
  - Works with any binary
  - Examples:
    - sample <pid> <duration>
    - sample myapp 10
  - Output in /tmp/<app>_<pid>.sample.txt
- ‘gdb attach’

Debugging With Project Builder

- Source level debugging
  - C, Obj-C, C++, and Java
- Full support for Mach-O binaries
  - Applications, frameworks, plug-ins, tools
- Uses gdb low-level Mac OS X debugger

Mac OS X Debugging Architecture



Session 706

Debugging With Project Builder

Rab Hagy
Project Builder Debugging Engineer

Project Builder Debugging: What You'll Learn

- Project Builder Debugging Status
- Configuring your project for debugging
  - Build settings
  - Executable settings
- Working with Breakpoints
- Debugging Java programs

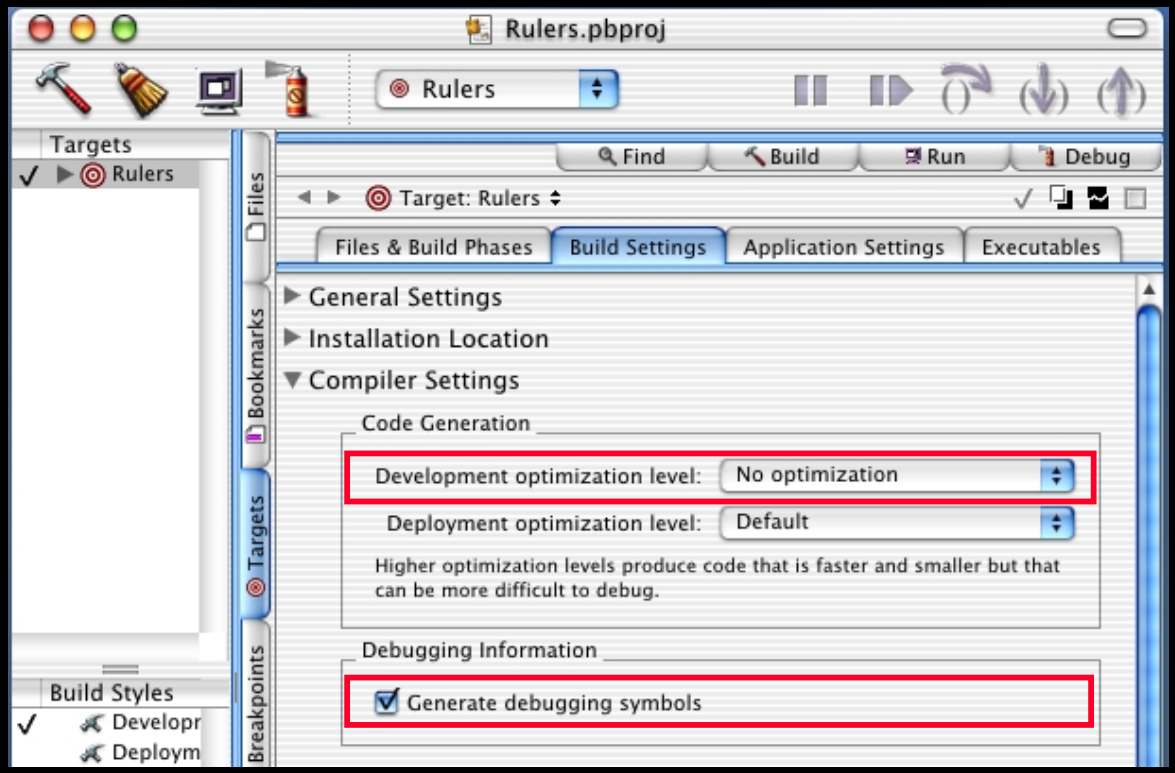
Project Builder Debugging Status

- Project Builder Debugger Features
  - GUI for threads, stacks, variables and PC
  - Breakpoints: persistent, source line, symbolic
  - Cross-project debugging
- Improvements since WWDC 2000
  - C++ support
  - PB/GDB communications
  - Java Debugging
- Still more to do…

Build for Debugging

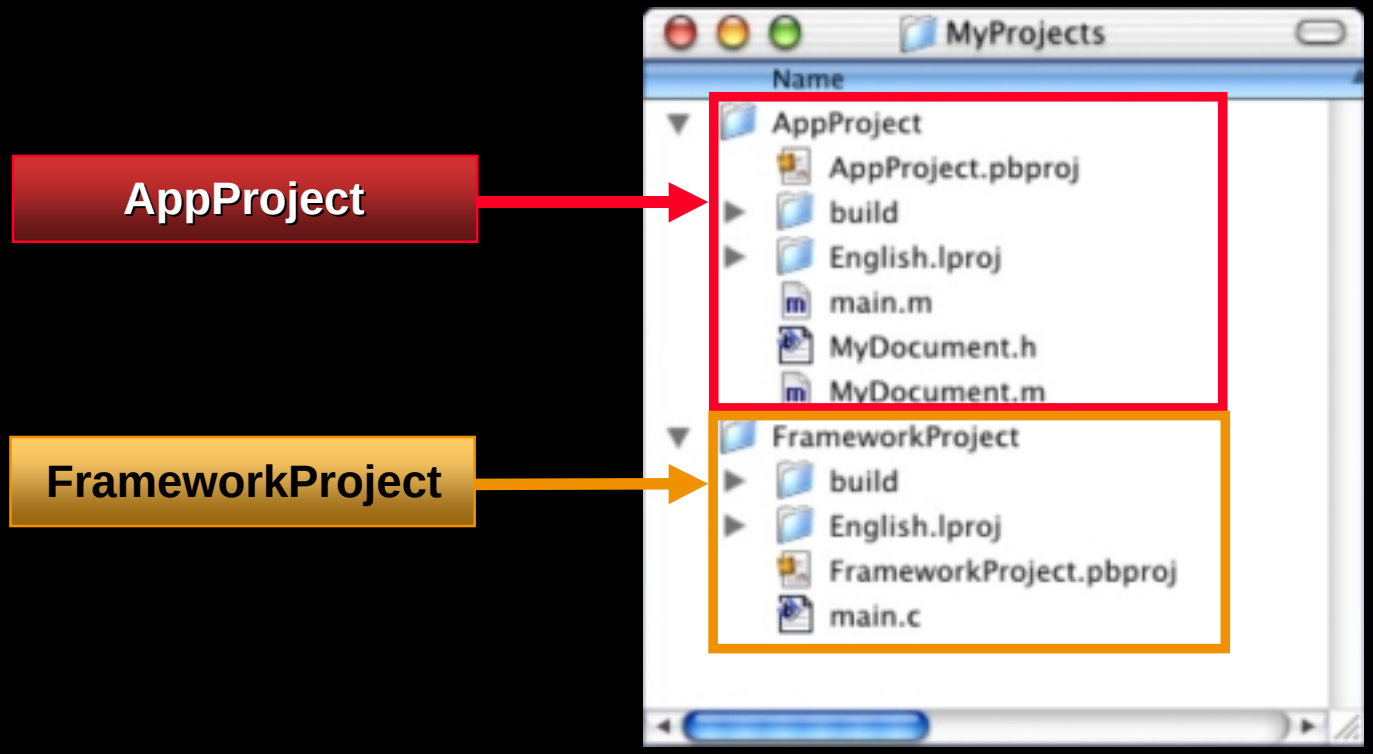
- Build Styles
  - Configure one Target for multiple uses
  - Development and Deployment
- Compiler Settings
  - Independent control of symbols, optimization
  - Symbols needed for source-level debugging
  - Optimization
    - None—default for Development Build Style
    - O1—OK for source-level debugging
    - O2—Sometimes OK, sometimes not
    - O3—Not recommended for source-level debugging

Target Build Settings



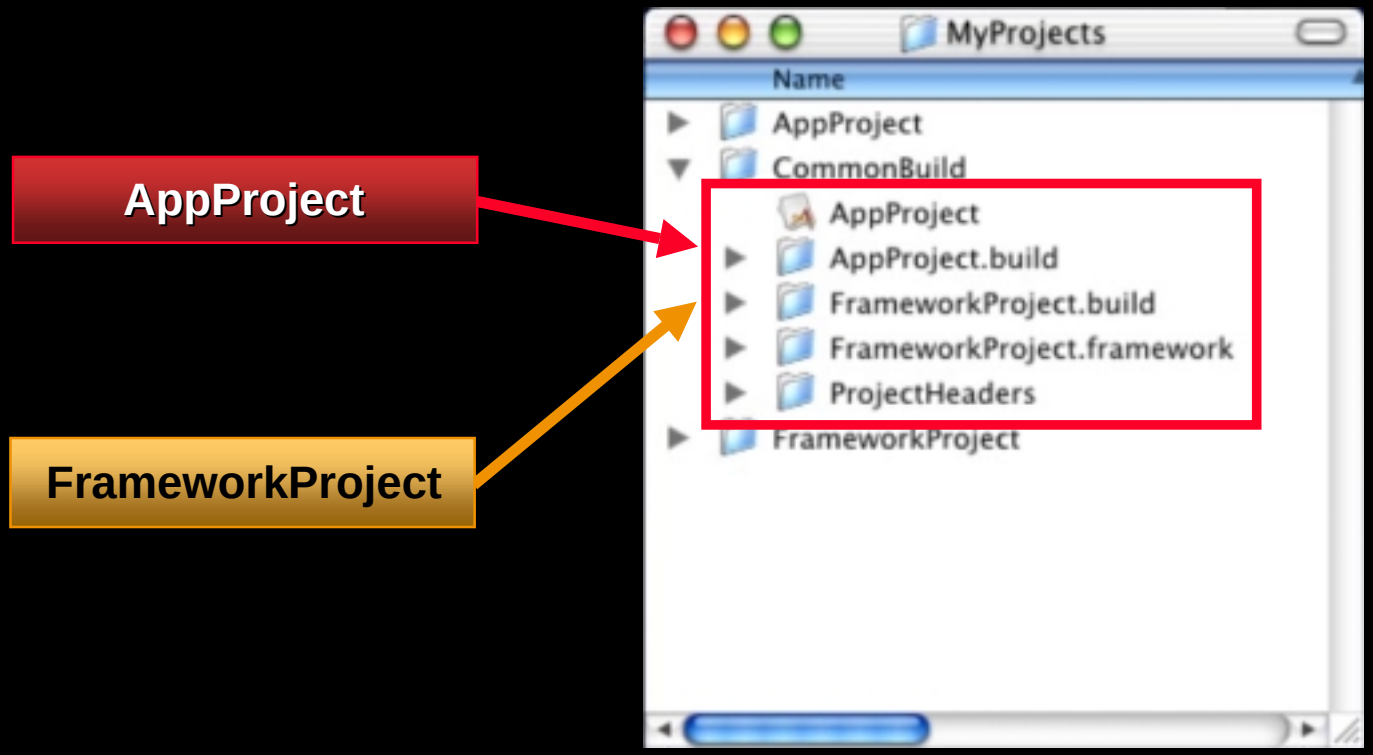
Building Related Projects

- Default is to build each project in own folder
  - Not suitable for related projects



Building Related Projects

- Build into a common folder
  - Solves build-time and runtime issues
  - "Project->Show Info" to set build folder



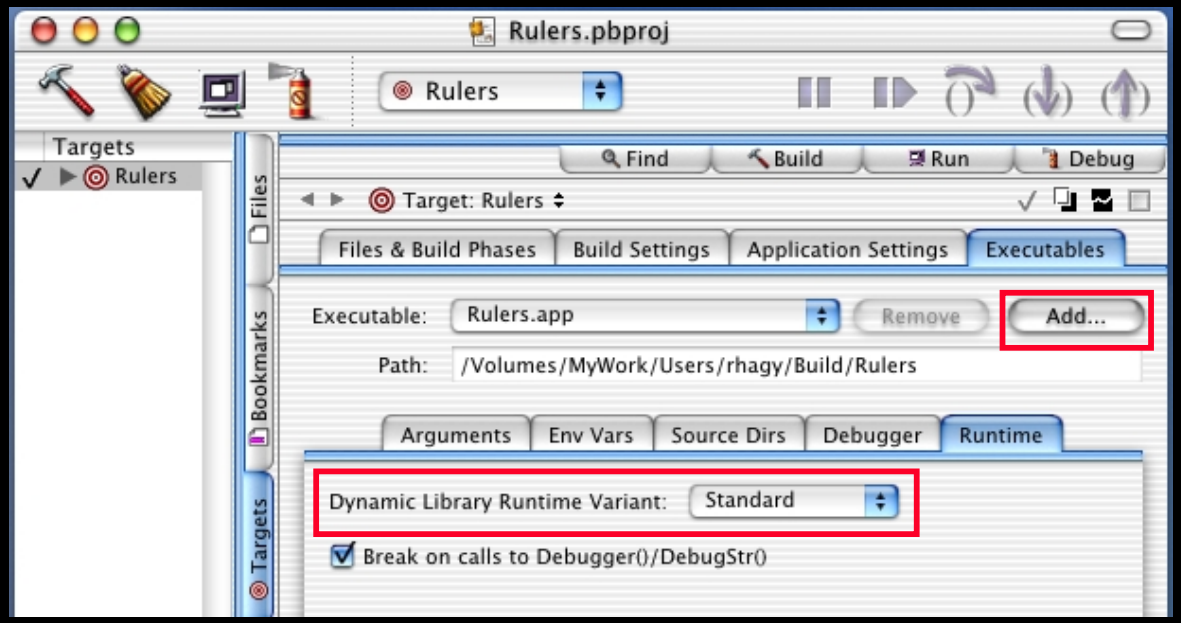
Executable Settings

- Executable
  - Program which is run or debugged
  - Selected by active target
  - Implicitly created by Application and Tool targets
  - Adding an additional Executable to target
    - External program, not under Project Builder
    - Library, Framework, Plug-in targets have no implicit executable
    - Program which loads your Plug-in

Executable Settings

- Arguments, environment
- System Frameworks variants
  - Standard: minimal symbols
  - Debug: additional error checking
  - Profile: profiling info

Executable Settings



Breakpoints

- Breakpoints tab
  - Summary of your breakpoints in project
  - Persistent, per-user, per-project breakpoints
  - Add button creates new symbolic breakpoint
- Breakpoints
  - In files
    - Click in margin to create, drag out to delete
  - Symbolic
    - In functions without full symbols
    - Library functions: open(), -[NSException raise]

Demo

Debugging a Cocoa Application

Session 706

Java Debugging

Dave Ewing
Project Builder Java Debugging Engineer

Java Debugging With Project Builder

- Setting a project up to debug Java
- Executables and command line arguments
- Java exceptions

Setting Up a Project for Java Debugging

- Turn on Indexing
  - Browsing information
  - Build dependencies
- Debugger source to class correspondence
  - Handles projects not laid out in package hierarchy
  - Faster Debugging (especially start up)

Java Executables

- Java Applications
  - MRJApp.properties
- Java Tools
  - Manifest file specifies Main-class
    - java -jar foo.jar
  - Class given on command line
    - java -classpath foo.jar Foo

Java Exceptions

- By default, debugger stops on uncaught exceptions
- Debugger doesn't know about JNI native methods that catch exceptions
- Java Console to specify other options
  - **catch none**
  - **catch uncaught**
  - **catch caught**
  - **catch all**

Demo

Java Debugging

Session 706

Advanced GDB Features

Jim Ingham
Senior Debugging Engineer

Debugging With GDB

- Goals
  - Developers don't need to know GDB
  - GDB users can use Project Builder debugging
- Why would you use GDB?
  - Good low-level debugger
  - Less common situations—kernel debugging…
  - Scriptable—breakpoint commands…
  - Extensible—user commands and plug-ins

Uncommon Debugging Situations

- Some situations will take longer to find their way into the UI
  - Remote debugging
  - PEF binaries
  - Kernel debugging

Remote Debugging

- Advantages
  - Can use debugger without deactivating App
  - Good for debugging full-screen applications
- Steps:
  - Start your application on the remote machine
  - telnet or ssh into remote machine
  - ‘attach’ to application
    - Use AppName<Tab> to look up PID…
- Project Builder support planned for future

GDB With Carbon PEF

- Mach-O is the preferred format for Mac OS X
- Can set breakpoints in Mach-O frameworks
- Steps
  - Build with TraceBack tables in CodeWarrior
  - **gdb <LongPath>/LaunchCFMApp**
  - **(gdb) run /Applications/MyApp**
- http://developer.apple.com/carbon/tipsandtricks.html

Kernel Extension Debugging

- Two-machine solution (KDP protocol)
  - Must be on same subnet
- Kernel looks like any other program
- Debugging agent in kernel talks to gdb
  - "Hard" kernel lock-ups can hang agent, causing gdb to lose connection to the kernel
- /Developer/Documentation/Kernel/Tutorials/KEXTutorials/index.html

Advanced Features

- Expression evaluator
- Breakpoint commands
- Conditional breakpoints
- User-defined commands
- Extension packages
- GDB manual is in the Developer package

Demo

Advanced Debugging

Klee Dienes
Senior Debugging Engineer

Expression Evaluator

- You can call any function in the target
- Functions CHANGE target state
- Syntax:
  - **call (void) DebugPrintWindowList ()**
  - or anywhere else you use a gdb expression

Breakpoint Commands

- You can run GDB commands at a breakpoint:

```
break CFRetain
commands
    set $count = (int) CFGetRetainCount ($r3) + 1
    printf "** retain 0x%lx; count = %d\n", $r3, $count
    where
end
```

- $r3 is the first argument to the called function

Breakpoint Commands

- You can continue GDB execution from within the breakpoint:

```
break CFRetain
commands
    set $count = (int) CFGetRetainCount ($r3) + 1
    printf "** retain 0x%lx; count = %d\n", $r3, $count
    where
    continue
end
```

Conditional Breakpoints

- You can tell GDB to stop only under certain conditions:

```
set $watch = str1
break CFRetain
condition $bpnum ($r3 == $watch)
```

- The expression passed to **cond** will be re-evaluated every time the breakpoint is hit
- The variable **$watch** is a “convenience” variable that holds a saved value

User-defined Commands

- You can add user-defined commands to GDB:

```
define print-counts
printf "str1 = %d\n", (int) GetRetainCount (str1)
printf "str2 = %d\n", (int) GetRetainCount (str2)
printf "str3 = %d\n", (int) GetRetainCount (str3)
end
```

Putting It All Together

- Now we can define user commands to watch a specific memory location for retains/releases:

```
define start-watching
    ...
end
define stop-watching
    ...
end
```

- Commands can be stored in a file and loaded using the **source** command in GDB

GDB Extensions

- You can also add C/C++/ObjC extensions:
  - Higher learning curve (can crash GDB)
  - See the GPL for redistribution terms.
  - Faster and more flexible than GDB commands
- Example: MacsBug
  - Available in **/usr/libexec/gdb/plugins/MacsBug**
  - See Developer website for more information and a minor (but important) bugfix

Summary

- Delivering on our commitment to great development tools
- Powerful debugging support in Project Builder
- Additional improvements on the way
- Please give us your feedback

Information Resources

- Information pages for all Mac OS X tools
  http://developer.apple.com/tools/

- Apple's mailing list master page
  http://lists.apple.com/

- Mailing lists:
  - **projectbuilder-users**
  - **carbon-dev**
  - **cocoa-dev**

Roadmap

FF119 Apple Developer Tools:
Feedback Forum

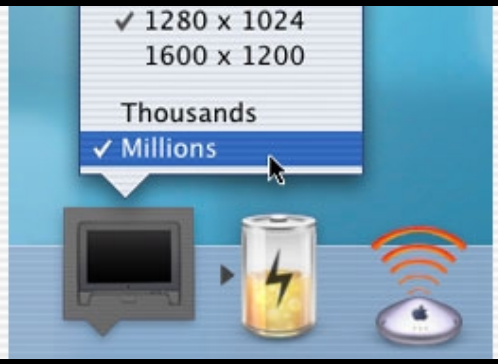
Room J1
Fri., 3:30pm

Who to Contact

Godfrey DiGiorgi
Technology Manager for Development Tools
Apple Worldwide Developer Relations
ramarren@apple.com

Macosx-tools-feedback@group.apple.com
Mailing address for feedback to Apple

Q&A



Godfrey DiGiorgi
Technology Manager for Development Tools
ramarren@apple.com

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001